ICT for Health Laboratory
Regression using Gaussian processes – Lab #2

Monica Visintin

Politecnico di Torino

November 4th 2022

Table of Contents

The experiment [1]

- A synthetic Gaussian random process $Y(t) \in \mathbb{R}$ is generated by passing white Gaussian noise through a Gaussian filter:

```
import numpy as np
import matplotlib.pyplot as plt
plt.close('all')
Np=200 # number of points in the Gaussian random process
Nm=21 # FIR filter memory
Nprev=2*Nm
T=10
th=np.arange(-Nprev,Nprev)
h=np.exp(-(th/T)**2) # Gaussian impulse response
h=h/np.linalg.norm(h)
x=np.random.randn(Np,) # input white Gaussian process
y=np.convolve(x,h,mode='same') # output filtered Gaussian process
t=np.arange(len(y))
```

The experiment [2]

- The impulse response of the filter and the realization of the random process are shown

```
plt.figure() # show the impulse response
plt.plot(th,h)
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('h(t)')
plt.title('Impulse response')
plt.show()
plt.figure() # show the realization
plt.grid()
plt.plot(t,y)
plt.xlabel('t (s)')
plt.ylabel('y(t)')
plt.title('Realization of the Gaussian random process')
plt.show()
```

The experiment [3]

- The theoretical autocorrelation function is evaluated and shown (no mathematical details on this):

```
autocorr=np.exp(-(th/T)**2/2)
plt.figure()
plt.plot(th,autocorr)
plt.xlabel(r'$\tau (s)$')
plt.ylabel(r'$R_Y(\tau)$')
plt.title('Autocorrelation function')
plt.grid()
plt.show()
```

The experiment [4]

- $M = 10$ points $\{[t_k, y(t_k)]\}_{k=1}^{10}$ are randomly selected:

```
M_sampled=10 # number of points used in GP regression
t_sampled=np.random.choice(t,(M_sampled,),replace=False)
y_sampled=y[t_sampled]
```

- Another time $t_\star$ is selected among the remaining times: for this $t_\star$ we want to perform Gaussian process regression, i.e. find $\hat{y}(t_\star)$ to be compared with $y(t_\star)$. Note that N in the slides is $M + 1 = 11$.

```
# randomly select the new point t_star for which we perform GP regression
t_rem=list(set(t)-set(t_sampled))
t_star=np.random.choice(t_rem,(1,),replace=False)[0]
y_true=y[t_star]
```

The experiment [5]

- At this point, it is necessary to find the covariance matrix $\mathbf{R}_Y$. If you had a vector of times $\mathbf{t} = [t_1, t_2, \ldots, t_N]$, you should first find the matrix with the time differences:

$$\mathbf{\Delta} = \begin{pmatrix} t_1 - t_1 & t_2 - t_1 & t_3 - t_1 & \cdots & t_N - t_1 \\ t_1 - t_2 & t_2 - t_2 & t_3 - t_2 & \cdots & t_N - t_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_1 - t_N & t_2 - t_N & t_3 - t_N & \cdots & t_N - t_N \end{pmatrix}$$

If `t` were a column vector, then it would be sufficient to write

```
1 delta_t_matr=t-t.T
```

However, in the example, `t` is/will probably be an array with shape $(N,)$, and it is necessary to make it a column vector first, and then evaluate $\mathbf{\Delta}$:

```
1 t_new=t[:, np.newaxis]
2 delta_t_matr=t_new-t_new.T
```

The experiment [6]

Then, to generate the covariance matrix, it is sufficent to write

```
R=np.exp(-(delta_t_matr/T)**2/2)
plt.matshow(R) # sow the covariance matrix
plt.colorbar()
plt.title('Theoretical covariance matrix')
```

- Note that in this specific case:
  - The covariance matrix mathematical expression is exactly known (we do not have to find the hyperparameters θ, r^2).
  - Since there is no measurement error, we set $\sigma_\nu^2 = 0$.
  - The regression model is simply

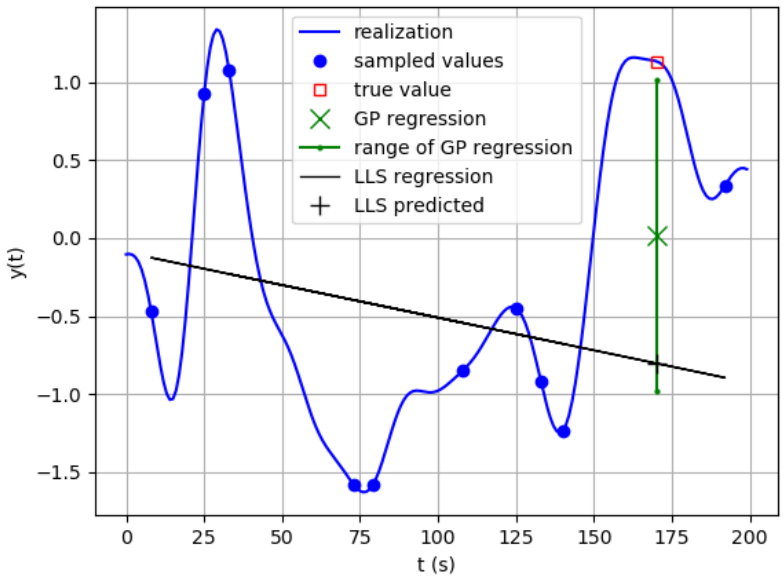
$$Y(t_\star) = g(t_\star)$$

  - You must be careful about **correctly picking the time values to be used to build the covariance matrix**.

To do [1]

- Correctly find $\mathbf{R}$.
- Find $\hat{y}(t_*)$ using the procedure given in the slides (slide 21).
- Find $\hat{y}(t_*)$ using LLS (Linear Least Squares).
- Plot
  - The realization.
  - The M points used as regressors (use for example marker `'o'`).
  - Point $[t, \hat{y}(t_*)]$ found through GP regression (use for example marker `'x'`).
  - The two points $[t, \hat{y}(t_*) + \sigma]$ and $[t, \hat{y}(t_*) - \sigma]$, where σ is the standard. deviation of the estimation in GP regression.
  - Point $[t, \hat{y}(t_*)]$ found through LLS (use for example marker `'+'`).

To do [2]



- Run your code several times (you'll see different results each time).

No report on this. The lines to find the GP regression will be used in the second part of the laboratory.

The overall picture

If you pick all the possible values of t_* (not just one at random), you get the following result:

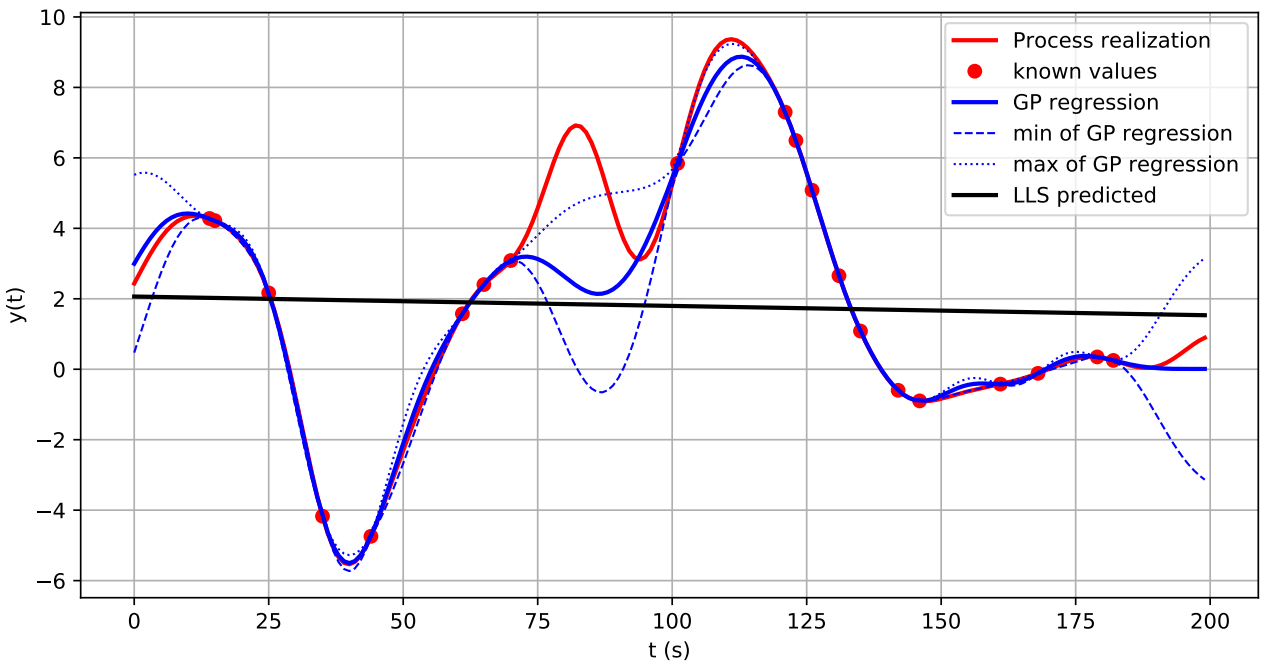


Table of Contents

Data preparation [1]

- As in Lab #1, load the Parkinson's disease dataset, remove the unwanted features, shuffle the data, get the training, and test datasets. To simplify, keep only the following features: total UPDRS (regressand), motor UPDRS, age, PPE (3 regressors).
- As a difference with respect to Lab # 1, the data must be divided into training, **validation** and test datasets according to the percentages: 50%, 25%, 25%.
- As in Lab # 1 find mean and standard deviation of each feature in the training dataset, and normalize the entire dataset using these means and standard deviations.
- As in Lab # 1 extract the regressand, i.e. total UPDRS, for the training, validation and test datasets.

Data preparation [2]

- The order of these 3 initial operations can be changed as you like, but the final result must be that you have a matrix `X_train_norm`, a vector `y_train_norm`, a matrix `X_test_norm`, a vector `y_test_norm`, a matrix `X_val_norm` a vector `y_val_norm`, all normalized, with obvious meanings. Use Ndarrays, not Pandas dataframes.

Goal [1]

We want to regress total UPDRS from the regressors (initially only motor UPDRS, age, PPE, later all the features used in Lab #1) using Gaussian processes:

$$Y = g(\mathbf{X}) + \nu$$

where $g(\mathbf{X})$ is the Gaussian process and ν the Gaussian measure error. The relevant formulas are the following:

- $N \times N$ covariance matrix $\mathbf{R}_{Y,N}$ in position n, k has value:

$$\mathbf{R}_{Y,N}(n,k) = \theta \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}_k\|^2}{2r^2}\right) + \sigma_\nu^2 \delta_{n,k}, \quad n, k \in [1, N]$$

- $N \times N$ covariance matrix $\mathbf{R}_{Y,N}$ is written as

$$\mathbf{R}_{Y,N} = \begin{bmatrix} \mathbf{R}_{Y,N-1} & \mathbf{k} \\ \mathbf{k}^T & d \end{bmatrix}$$

Goal [2]

- The pdf of Y_N for regressors $X = x_N$, given the measured values $\mathbf{y} = [y_1, \ldots, y_{N-1}]$ is

$$f_{Y_N|\mathbf{y}}(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

$$\mu = \hat{y}_N = \mathbf{k}^T \mathbf{R}_{Y,N-1}^{-1} \mathbf{y}$$

$$\sigma^2 = d - \mathbf{k}^T \mathbf{R}_{Y,N-1}^{-1} \mathbf{k}$$

- In the above equations, couples $(\mathbf{x}_n, y_n)$ for $n = 1, \ldots, N-1$ belong to the training dataset, couple $(\mathbf{x}_N, Y_N)$ belongs to the test or validation datasets. Note that Y_N is the random variable whose pdf we want to estimate, y_N is its true value, $\hat{y}_N$ is its estimated value.
- **Hyperparameters θ, r, σ_ν^2 must be found using the validation dataset, i.e. using $(\mathbf{x}_N, Y_N)$ from the validation dataset.**

Initial procedure to check that the system works [1]

In the model, initially set the autocorrelation hyperparameters $r^2 = 3$, $\theta = 1$ and $\sigma_\nu^2 = 0.001$. Moreover, set in Python `N=10` (`N` has the same meaning as N in the slides)

1. Pick all the rows of `X_val` and perform the following steps for each of the rows. Let `x` be the k-th row of `X_val_norm` with normalized UPDRS equal to `y_val_norm[k]`

```
1    x=X_val_norm[k,:]
```

2. Find the $N-1$ points in the training dataset that are closer to `x`.
3. Build the covariance matrix $\mathbf{R}_N$.
4. Find the estimated normalized value of total UPDRS for `x` using GP regression. Exploit the code you wrote in the first part of this laboratory.
5. Generate the (normalized) UPDRS value corresponding to `x`, i.e. the estimate of `y_val_norm[k]`, using the formula of μ in the slides.
6. Verify that you get reasonable results by plotting, as usual, the estimate of `y_val_norm` versus `y_val_norm` (regression line).

Procedure to set the hyerparameters [1]

- Note that, since the dataset was normalized, the autocorrelation $R_Y(\mathbf{0})$ (i.e. the values on the main diagonal of the covariance matrix) are equal to 1 and it is not necessary to optimize this value (i.e. $\theta = 1$).
- Parameters to be set are: `r2` and `s2`. They must be set so that **the mean square value of `y_val-yhat_val` (i.e. the validation mean square error) is minimized**. Equivalently you can minimize the mean square value of `y_val_norm-yhat_val_norm`. Use a set of values for `r2`, a set of values for `s2`, measure the validation mean square error, find the minimum (this procedure is called **grid search**).
- Don't ask the instructor about the range that you must consider for `r2` and `s2`, use common sense, find a solution, you are a master student.
- Moreover you should also have to correctly set `N`: we cannot use `N` equal to the number of datapoints in the training set (the covariance matrix would be too large), but `N=10` might be too small. However, use $N = 10$, it gives reasonable results.

Procedure to set the hyerparameters [2]

- You should also try and see what happens if the features used for this regression are not just motor UPDRS, age and PPE, but keep only these regressors, they give reasonable results.

Procedure to set the hyerparameters [3]

- In the overall you must find reasonable values of the hyperparameters

  `r2, s2`.

  keeping $N = 10$ and the suggested regressors.

Testing

Once `r2, s2` have been optimized for the validation dataset, measure the following for the **test dataset**

1. the mean, the standard deviation, the mean square value of the estimation error for the un-normalized total UPDRS
2. the histogram of the un-normalized estimation error
3. the plot of the estimated total UPDRS versus the true one with the errorbars (use 3 times the value of `sigma` generated in line 19 of the code in the previous slides, but remember to un-normalize this value)
4. the value of R^2

so that you can compare the results obtained with the Gaussian process regression with the results obtained with linear regression LLS in Lab #1. Note that each time you run your code you might get different results, if shuffling of the data is random.
No report due.